

A Developer's Diary: The Trials and Triumphs of Maintaining Devboards in AOSP



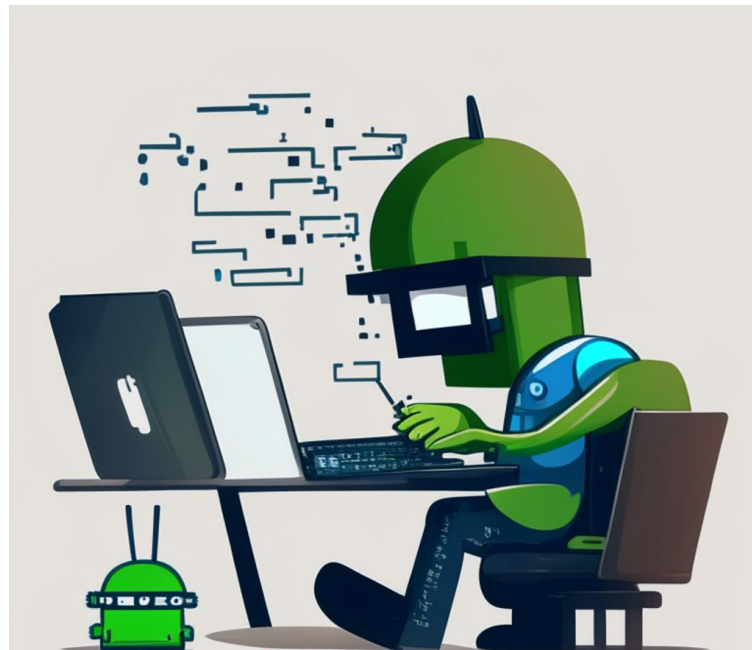
A Developer's Diary: The Trials and Triumphs of Maintaining Devboards in AOSP

Product names, logos, brands, products and other trademarks featured or referred to within this document are the property of their respective trademark holders.



About Me!

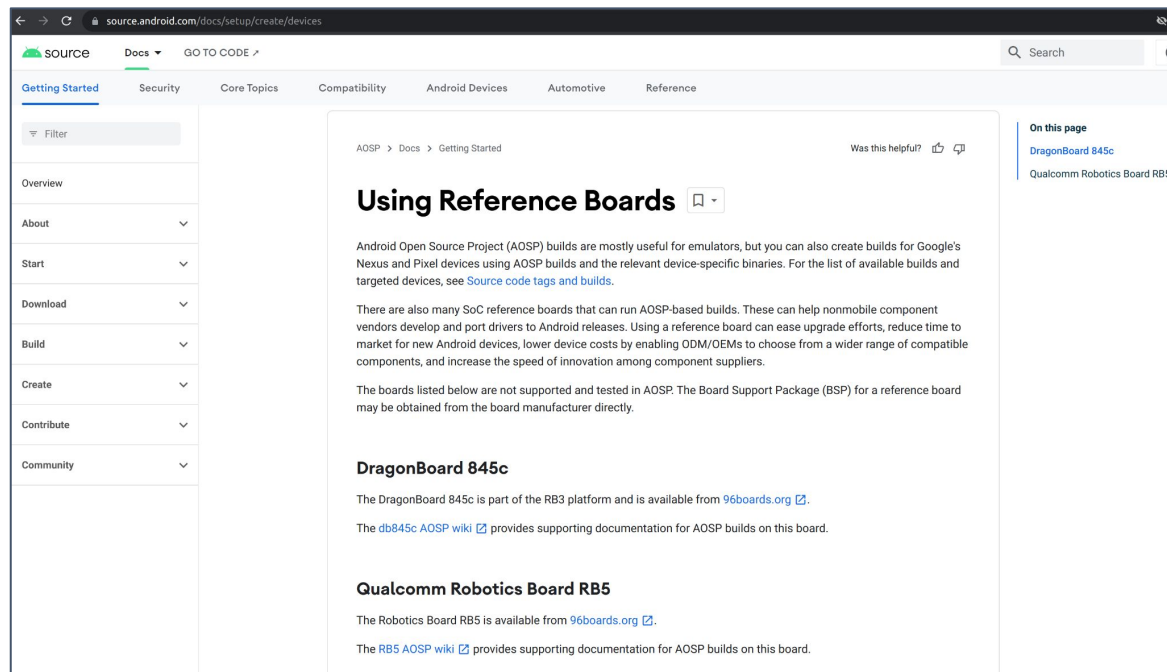
- Senior Engineer at Linaro
- 10+ years of AOSP bringup and maintenance on devboards
- pundir on #linaro-android, #aosp-developers IRC channels at OFTC.net



Agenda!

- Devboards / Reference boards in the **Android Open Source Project (AOSP)**
- Linaro, Devboards and the AOSP
- Keeping up with the AOSP
- Tracking relevant upstream projects
- Supporting various device build configurations

Devboards or Reference Boards in AOSP



The screenshot shows the source.android.com/docs/setup/create/devices page. The left sidebar contains a navigation menu with links: Overview, About, Start, Download, Build, Create, Contribute, and Community. The main content area is titled 'Using Reference Boards' and includes the following text:

Android Open Source Project (AOSP) builds are mostly useful for emulators, but you can also create builds for Google's Nexus and Pixel devices using AOSP builds and the relevant device-specific binaries. For the list of available builds and targeted devices, see [Source code tags and builds](#).

There are also many SoC reference boards that can run AOSP-based builds. These can help nonmobile component vendors develop and port drivers to Android releases. Using a reference board can ease upgrade efforts, reduce time to market for new Android devices, lower device costs by enabling ODM/OEMs to choose from a wider range of compatible components, and increase the speed of innovation among component suppliers.

The boards listed below are not supported and tested in AOSP. The Board Support Package (BSP) for a reference board may be obtained from the board manufacturer directly.

DragonBoard 845c

The DragonBoard 845c is part of the RB3 platform and is available from [96boards.org](#).

The [db845c AOSP wiki](#) provides supporting documentation for AOSP builds on this board.

Qualcomm Robotics Board RB5

The Robotics Board RB5 is available from [96boards.org](#).

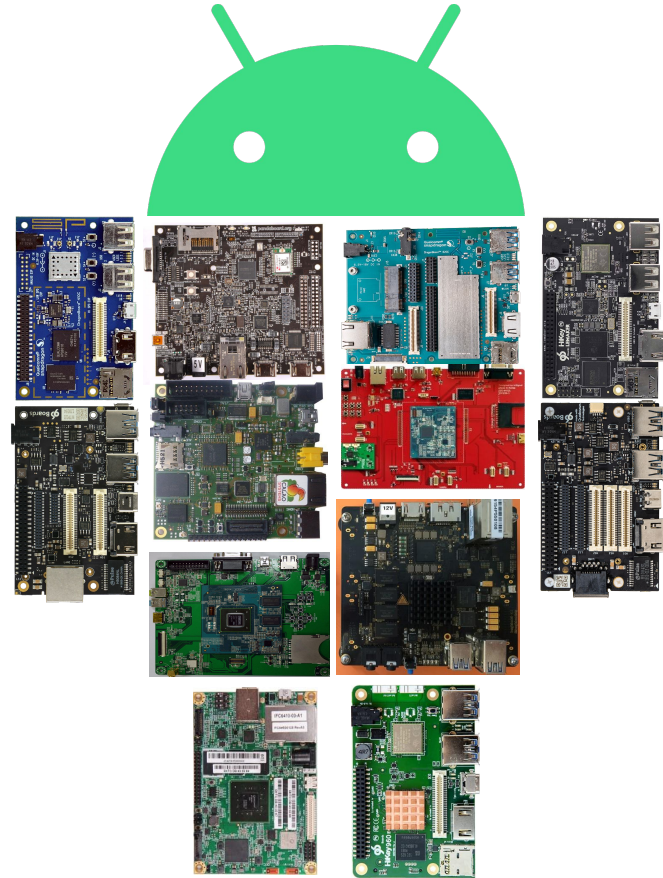
The [RB5 AOSP wiki](#) provides supporting documentation for AOSP builds on this board.

On the right side of the page, there is a 'Was this helpful?' section with thumbs up and thumbs down icons, and a 'On this page' section listing 'DragonBoard 845c' and 'Qualcomm Robotics Board RB5'.

Why Devboards in AOSP?

- Combines latest upstream projects with latest AOSP
 - Creates space of overlapped interests between Android and upstream projects
- Vehicle for testing for catching and fixing regressions
 - Focussed on core generic system (kernel, graphics, connectivity) functionality
 - Multiple -stable releases & Mainline
 - Critical for vendor confidence in updating kernels by comparing upstream and downstream solutions
- Development and prototyping of new functionality for AOSP
 - Feature (GKI / AVF / ..) prototyping
 - Development of generic HALs utilizing modern upstream kernel interfaces

Linaro, Devboards and Android Open Source Project



Linaro Devboards and AOSP

- Linaro support AOSP on a variety of member development boards
- These devboards are chosen based on a number of factors like,
 - **Relevance to AOSP**
 - Near flagship h/w specs to keep up with ever increasing AOSP requirements
 - Outside of spec requirements, some boards have been interesting for the specific features they provide.
 - For example: BeagleBoard-X15 (32bit platform) and Hikey960 (brought in big.LITTLE)
 - **Upstream support**
 - Helps deliver software and security updates
 - **Active community and documentation**
 - Limited user base or lack of active community support makes it challenging to find solutions to specific problems.
 - Incomplete or outdated documentation makes it difficult to set up and configure AOSP properly.
 - **Popularity and Availability of device**

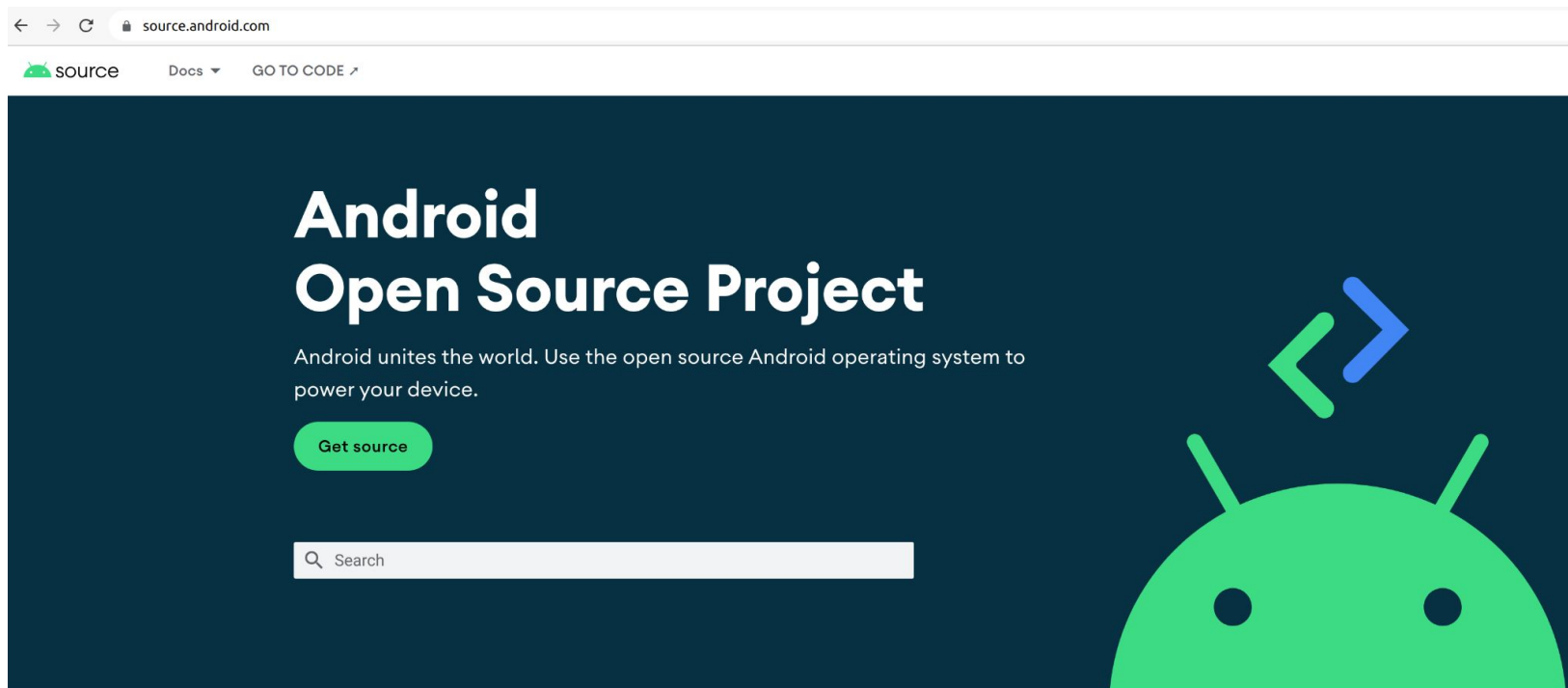
Linaro Devboards and AOSP

- We ensure that latest AOSP builds can be installed and run on supported devboards with latest LTS kernel versions.
- A well supported devboard can help to reduce development time and costs
 - Developers can be confident that the software they are developing will run smoothly on the chosen hardware.
- Perform extensive testing coverage, including both functional and performance testing, as well as CTS and VTS compatibility test suits.

Linaro Devboards and AOSP

- Testing AOSP with **Linux Kernel Quality** project and **Linux Kernel Functional Testing**
- Ran **~470 M** Android tests since Jan'23, **~2.43 B** since the start
- We test **13** kernel variants across **4** userspace combinations in LKFT using CTS and VTS test suits
 - Kernels: android-mainline, android14-6.1, android14-5.15, android13-5.15, android13-5.10, android12-5.10, android12-5.10-lts, android12-5.4, android12-5.4-lts, android11-5.4, android11-5.4-lts, android-4.19-stable, and android-4.14-stable
 - Userspace: AOSP, Android13, Android12, and Android11
- Helped fix **40** regressions since Jan'23, on all dev-boards, resulting in **6** upstream and **22** AOSP fixes.

Keeping up with AOSP



Choosing the Right Droid

The **Android Release** tag and the **Android Open Source Project (AOSP)** are related but distinct concepts in the Android ecosystem.

- An Android release tag refers to a specific version of the Android operating system
 - Android release represent major updates to the Android platform and include new features, bug fixes, and security enhancements.
 - Serve as a reference for developers to target specific versions of Android when building apps.
 - Guarantees stable interfaces via -gsi branches suitable for app developers.

Choosing the Right Droid

- Unlike Android release tag code drops, AOSP is an active development project.
 - AOSP is the foundation upon which future Android releases are build.
 - Usually new features land in AOSP before they get shipped in future releases.
 - Making the device future ready, to avoid any surprises during next release drop.
 - Being an actively development project, it is more prone to regressions.
 - Limited vendor support.
 - Eventually the source code for each Android version is made available through AOSP.

Keeping up with AOSP

When using the Android Open Source Project (AOSP) for your project, there are several common pitfalls that we encounter.

- Not a good time/day/week to sync AOSP sources
 - Transient build or runtime failures due to async project updates
 - Caught up in between major framework changes
 - repo superprojects to the rescue?
- Framework changes that need device configuration changes
 - For example: HIDL to AIDL transition
 - Cuttlefish to the rescue!
- Keeping up with .mk / .bp / .bazel build configurations

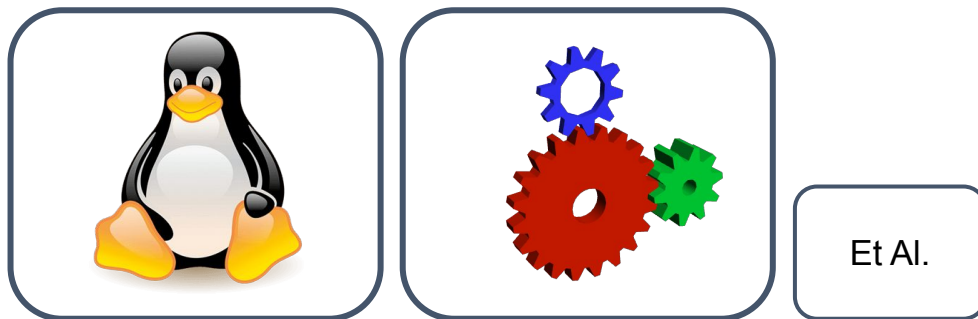
Keeping up with AOSP

- Feature dependency on Android Common Kernel patch(es)
 - Boot dependency on Ashmem
 - Transition to memfd is still WIP
 - Remount dependency on overlayfs patches
 - /metadata encryption on dm-default-key
- For ever changing Boot Image Header version requirements
 - Vendor bootloader binaries barely get feature updates
 - Makes it difficult to update to latest features

Keeping up with AOSP

- Troubleshooting Generic Kernel Image (GKI) boot failures
 - pre-KABI (Kernel-ABI) freeze miseries
 - Symbol list modifications
 - Removal of exported symbols can break GKI for vendor drivers
 - Shipping protected kernel modules
 - Vendors aren't allowed to update protected kernel modules
 - GKI config changes
 - For example, removal of FW_LOADER_USER_HELPER_FALLBACK broke firmware loading for a few drivers. So we end up shipping a copy of f/w binaries in vendor ramdisk as a workaround.

Tracking Relevant Upstream Projects



Tracking Relevant Upstream Projects (Linux)

Working with the upstream Linux kernel (or LTS) and AOSP bring long term benefits, but it can be a pain at times.

- DT nodes / sysfs entries do not count as stable interfaces
 - Device peripherals being probed thru device files are often a pain with SELinux's access control lockdowns.
- If it is not reproducible on Debian/Fedora/OE then is it even a valid bug?
- If the bug report is tied to a firmware bug then good luck with that
 - Look for a userspace workaround
 - Gets worse if it is an OEM signed firmware
- Some upstream regression reports just go unanswered
 - Hitting a kernel warning during Hostapd teardown for example

Tracking Relevant Upstream Projects (Open Graphics)

Tracking open-source graphics (viz **Mesa**, **libdrm**, **drm_hwcomposer**, **minigbm**) upstream-main projects mostly in the Freedreno driver context.

- Integrating Freedreno into AOSP enhances the compatibility of Android with devices that utilize Qualcomm Adreno GPUs.
 - Help ensure proper graphics acceleration, stability, and performance on a wider range of devices.
 - Increase the lifetime of the device, even beyond the original device manufacturer's support lifecycle.

Tracking Relevant Upstream Projects (Open Graphics)

Although it provides a number of benefits and advantages, these are potential pain points of using Mesa/Freedreno based open-graphics stack on AOSP.

- Fairly regular upstream Mesa build breakages with AOSP
 - Non-compatible / older version of python-3 or expat on the Host machine for example
 - Or a supporting package being dropped from AOSP (libbacktrace an example)
- One particular upstream Mesa runtime crash worth mentioning, where Mesa builds with AOSP was crashing at runtime but working fine when built with android13-gsi branch
 - Reported the breakage upstream
 - Ended up shipping android13-gsi built Mesa binaries on AOSP builds

Tracking Relevant Upstream Projects (Open Graphics)

- **AOSP/external/mesa3d** project is awfully outdated and updating it to the latest stable version is a non-trivial exercise
 - We ended up shipping upstream Mesa prebuilt binaries for our devices
- In this one particular usecase, fixing upstream project was trivial but merging the changes back in AOSP turned out to be more painful exercise
 - Because it broke Cuttlefish's no runtime SELinux denial rule during merge presubmit tests

Supporting Various Device Build Configurations



Supporting Various Device Build Configurations

AOSP bringup on newer or pre-silicon or resource constraint SoCs can be fun and challenging exercise but a great learning experience nevertheless.

- Be it booting minimal rootfs with only console (adb shell)
- Preparing a generic target build using software rendering in case of missing GPU support
 - Using ANGLE as GLES implementation on top of SwiftShader's vulkan implementation (Pastel)
- One build target to rule them all (well most of them)
 - Getting unified AOSP boot images booting on devboards with different SoCs but from the same Silicon vendor

Summary

- Maintaining AOSP on devboards **can be a complex and challenging** process. Starting from choosing the right devboard to dealing with software configuration and customization challenges, lack of support or limitations of the devboard itself.
- But, using devboards for AOSP development **also offers a range of advantages**, including hardware compatibility testing, rapid prototyping of new features, system integration, optimization and debugging capabilities, and cost-effective development options.
- Overall, devboards **have often proven to be essential tools** in the AOSP development process and **serving as reference platforms** for Android device manufacturers.

Questions?

Thank you

Amit Pundir <amit.pundir@linaro.org>

IRC: pundir on #linaro-android, #aosp-developers @OFTC

EOSS | ELC Prague 2023, 26-30 June

